PATENT ABSTRACTS OF JAPAN

(11)Publication number : **2001-357048A**

(43)Date of publication of application : **26.12.2001**

(51)Int.Cl. **G06F 17/30**

G06F 7/24

G06F 12/00

H03M 7/30

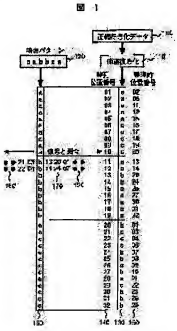
(21)Application number : **2000-182320**

(22)Date of filing : **13.06.2000**

(71)Applicant : **HITACHI LTD**

(72)Inventor : **TONOMURA MOTONOBU**

(54) METHOD FOR RETRIEVING BLOCK SORT COMPRESSED DATA AND ENCODING METHOD FOR BLOCK SORT COMPRESSION SUITABLE FOR RETRIEVAL



(57)Abstract:

PROBLEM TO BE SOLVED: To perform a high-speed retrieval by successively decoding and retrieving only required data even without decoding all encoded data concerning data compressed by a block sort compression method.

SOLUTION: Concerning data compressed by the block sort compression method, a pair of the alignment position number and the pre-alignment position number of a BW-converted column and columns aligned in a dictionary form is found. Then, collation with a retrieval data stream is performed while decoding the data on the basis of that pair. Decoding is applied only to a part required for retrieval. Besides, a pair of the alignment position number and the pre-alignment position number is directly encoded by the block sort compression method.

D3

(19)日本国特許庁（ＪＰ）　(12) **公開特許公報（Ａ）**　(11)特許出願公開番号
特開2001－357048
（P2001－357048A）
(43)公開日　平成13年12月26日(2001.12.26)

(51)Int.Cl.[7]	識別記号	ＦＩ		ﾃｰﾏｺｰﾄﾞ(参考)
Ｇ０６Ｆ　17/30	２３０	Ｇ０６Ｆ　17/30	２３０Ａ	５Ｂ０７５
	１７０		１７０Ａ	５Ｂ０８２
7/24		7/24	Ｚ	５Ｊ０６４
12/00	５１１	12/00	５１１Ｃ	
Ｈ０３Ｍ　7/30		Ｈ０３Ｍ　7/30	Ｚ	

審査請求　未請求　請求項の数6　ＯＬ　（全 11 頁）

(21)出願番号　特願2000－182320(P2000－182320)
(22)出願日　平成12年6月13日(2000.6.13)

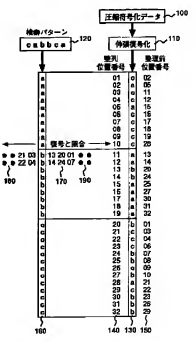
(71)出願人　000005108
株式会社日立製作所
東京都千代田区神田駿河台四丁目６番地
(72)発明者　外村　元伸
東京都国分寺市東恋ヶ窪一丁目280番地
株式会社日立製作所中央研究所内
(74)代理人　100068504
弁理士　小川　勝男　（外２名）

Ｆターム(参考)　5B075 ND02 NR16 PQ02
5B082 AA11 GA02
5J064 AA02 BC29 BD04

(54)【発明の名称】　ブロックソート圧縮データの検索方法、および検索に適したブロックソート圧縮法の符号化方法

(57)【要約】
【課題】ブロックソート圧縮法で圧縮されたデータに対して、全ての符号化されたデータを復号しなくても、逐次必要なデータだけ復号して検索することによって高速検索をおこなう。
【解決手段】ブロックソート圧縮法により圧縮されたデータに対して、ＢＷ変換した列と辞書式に整列させた列との整列位置番号と整列前位置番号の対を求める。そして、その対に基づきデータを復号しながら、検索データ列との照合をおこなう。復号は、検索のために必要な部分のみをおこなう。また、ブロックソート圧縮法で整列位置番号と整列前位置番号の対を直接に符号化しておく。

図 1



1

【特許請求の範囲】

【請求項１】　ブロックソート圧縮データの検索方法において、

ブロックソート圧縮法により符号化されたデータの列を第一の列、その第一の列を辞書式に整列させたデータの列を第二の列としたとき

（１）前記第一の列のデータを前記第二の列のデータに整列させたときの整列位置番号と整列前位置番号の対を求めるステップ、

（２）前記（１）のステップにより求められた整列位置番号と整列前位置番号の対に基づき、元のデータ列を復号するステップ、

（３）検索データ列を前記（２）のステップにより復号されたデータ列を照合するステップとからなり、

前記第一の列のデータと検索データ列を入力し、

前記（１）のステップの後に、前記（２）のステップをおこない、順次復号されたデータから前記（３）のステップをおこなって元のデータ列に検索データ列が含まれるか否かを調べることを特徴とするブロックソート圧縮データの検索方法。

【請求項２】　ブロックソート圧縮法により符号化されたデータは、データの構成要素の各出現個数がわかるように、符号化されており、

前記（２）のステップにおいて、全ての元のデータ列を復号しなくても、

前記データの構成要素の各出現個数に基づいて、前記整列位置番号と整列前位置番号の対を求めて、前記（３）のステップで検索データ列との照合に必要なデータだけ復号して検索をおこなうことを特徴とする請求項１記載のブロックソート圧縮データの検索方法。

【請求項３】　ブロックソート圧縮法の符号化方法において、

巡回シフトの後に、取り出した列を符号化するときに、その取り出した列を、辞書式に整列した列に変換する場合の整列位置番号と整列前位置番号の対を直接に符号化することを特徴とするブロックソート圧縮法の符号化方法。

【請求項４】　前記（３）のステップと検索データ列と元のデータ列を照合するときに、元のデータ列の構成要素の出現個数のすくない構成要素から照合をおこなうことを特徴とする請求項１記載のブロックソート圧縮データの検索方法。

【請求項５】　前記検索データ列が一意的でない構成要素の表現がされ、

前記（３）のステップと検索データ列と元の列を照合するときに、その表現により、複数の構成要素とマッチングをすることにして検索することを特徴とする請求項１記載のブロックソート圧縮データの検索方法。

【請求項６】　前記（３）のステップにおいて、

元のデータ列の中で、前記検索したデータ列を含まれる

2

場所の前後のデータ列をも、復号して表示することを特徴とする請求項１記載のブロックソート圧縮データの検索方法。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】本発明は、ブロックソート圧縮データの検索方法に係り、ブロックソート圧縮法で圧縮されたデータに対して、ブロックソート圧縮法の特質を活かすことにより、全ての符号化されたデータを復号しなくても、逐次必要なデータだけ復号して検索することによって高速検索を可能にするブロックソート圧縮データの検索方法に関する。

【０００２】

【従来の技術】コンピュータなどの情報機器が身近なものになり、我々がデジタルデータを取扱う機会が増すにつれて、データを符号化して圧縮して記憶し、必要に応じて、それを復号化して伸張して利用する技術が注目されている。ここで、「符号化」とは、もとのコード体系を別のコード体系に変換することであり、「復号化」とは、その逆であり、符号化したコード体系をもとのコード体系に変換し直すことと定義できる。また、「圧縮」とは、もとのデータをより少ない容量の記憶領域で格納することであり、「伸張」とは、圧縮したデータをもとのデータの容量の記憶領域を占有するようなデータにすることである。

【０００３】このようなデータの圧縮／伸張技術は、パーソナルコンピュータが普及して、日常的に使われるようになってきている。中でも、１９７７年にZivとLempelによって提案されたＬＺ圧縮法は、知名度があり今日よく使われている圧縮法である。このような状況の中で、最近、ＬＺ圧縮法の圧縮率に匹敵するブロックソート圧縮法と呼ばれる別の圧縮法がその圧縮率の高さから理論的な面で関心が集まっている(Michelle Effros, Universal Lossless Source Coding with the Burrows Wheeler Transform, IEEE Proc. of DCC'99, pp. 178-187, 1999)。

【０００４】このブロックソート圧縮法は、テキスト・データ全体に対して巡回シフト(あるいは回転シフト)列を作り、すべての巡回シフト列に辞書式順序付けをおこなって配列し、そのある列を取り出して符号化するものである。例えば、この方式を発案したBurrowsとWheeler (A block-sorting lossless data compression algorithm, SRC Research Report, 124 May 1994)は、符号の列として配列の最後の列を選んでいる。

【０００５】ブロックソート圧縮法は、ＬＺ圧縮法に匹敵するような圧縮率の評価データが得られているが、まだその圧縮率の達成に関して理論的な検討段階にあり、応用面での検討があまりなされていない。

【０００６】【発明が解決しようとする課題】ところで、情報を圧縮

するばかりでなく、情報を高速に検索したいという要求がある。高速検索という目的を優先させると、どうしても検索のために冗長な情報をもたせることになるため、データ量を圧縮するというよりは、ともすれば、データ量が増加することにもなる。

【０００７】しかし、処理する情報が極めて大量になってくると格納スペースがなくなり、データを圧縮格納しておく必要に迫られるので、ほとんどのデータが圧縮されたままの状態で眠っているという状況になる。このような圧縮データの海から検索して必要なごく一部のデータのみを取り出すという技術が要求されることになる。すべての圧縮データを伸張復号しながら検索するということは現実的な解答とは言い難い。そのために、できれば圧縮コード列のまま伸張しないで検索する方法を見い出したいところである。

【０００８】しかしながら実際には、既存のＬＺ圧縮法によるデータに対して、圧縮コード列に対して検索パターンと比較照合する場合、圧縮コード内容の前後の一部にまたがって照合したり、検索パターンの圧縮コード列が一意に定まらず、何通りも存在することがあるため、圧縮コード列のままでは直接検索することができないという問題がある。ブロックソート圧縮法は、評価段階の圧縮技術と言って良く、これまで、ブロックソート圧縮データに対する検索法について十分考察されてこなかった。

【０００９】本発明は、上記問題点を解決するためになされたもので、その目的は、ブロックソート圧縮法で圧縮されたデータに対してブロックソート圧縮法の特質を活かすことにより、全ての符号化されたデータを復号しなくても、逐次必要なデータだけ復号して検索することによって高速検索を可能にするブロックソート圧縮データ検索法を提供することにある。

【００１０】また、その検索に適したブロックソート圧縮法の符号化方法を提供することにある。

【００１１】

【課題を解決するための手段】ブロックソート圧縮法は、テキスト・データ全体に対して巡回シフト列を作り、すべての巡回シフト列に辞書式順序付けをおこなって配列化する。それで、検索したいパターンが複数箇所にある場合、巡回シフト列の配列の中で、検索パターンは配列のある行の先頭列から始まり、しかも複数箇所の検索パターンが配列の連続した行において固まって出現するという特徴がある。また、ブロックソート圧縮データの復号原理において、伸張復号した最後の列の文字列の位置を辞書式順序で並べ替えて整列させるが、そのとき、整列位置番号と整列前の整列位置番号の対を作り、元のテキストの整列位置番号を指定して、これらの対を順次たどりながら、テキストの先頭から順番に復号できる。

【００１２】したがって、本発明では、ブロックソート圧縮データがもつこのような性質を利用して効率良い検索手段を提供する。すなわち、まず検索パターンの先頭の文字と２番目の文字の対に対して、整列位置番号と整列前位置番号の対を当てはめて対応させる。これらに当てはめられる対は、辞書式順序で整列させているために、固まって出現するので候補を絞り込める。続いて検索パターンの２番目の文字と３番目の文字の対に対して、前段で絞り込まれたものに対して整列位置番号と整列前の整列位置番号の対を当てはめて対応させるという手順を同様に繰り返していく。すると、検索パターンの長さをｎとするとき、検索パターンのｎ－１番目の文字とｎ番目の文字の対に対して整列位置番号と整列前位置番号の対を当てはめて絞り込み対応させた段階で一連の手順は終了する。

【００１３】結果として、複数箇所で検出された検索パターンだけが残り、同時に元のデータ列に含まれている検索パータンが検出されることになる。

【００１４】元のテキスト列の文字の出現個数がわかっているときには、整列位置番号と整列前位置番号の対を順次求めることができるため、照合にあたり全ての元のテキスト列を復号する必要はなく、検索パターンと照合する必要なところだけ復号して照合すれば良い。また、いわゆるあいまい検索も、マッチングの可能性のあるところを上記手順で復号することによりおこなうことができる。

【００１５】検索パターンを照合するときには、元のテキスト列で出てくる文字が一番少ない文字からおこなえば絞込みが速くなり、効率的に検索できる。

【００１６】ブロックソート圧縮法では、符号化操作は２段階でおこなわれる。第１段階の符号化では、通常の発想では、連続して出てくる文字の長さに着目して符号化するものである。しかしながら、上記検索方法では、第１段階の復号化のステップと整列位置番号と整列前位置番号の対を求める手順が無関係に存在しており、効率的には改善すべきものがある。

【００１７】そこで、ブロックソート圧縮法において、配列の最後の列の文字列を圧縮符号化するのではなく、整列位置番号と整列前位置番号の対を直接に圧縮符号化することで復号と検索の効率をさらに上げることにする。整列位置番号と整列前位置番号の対は、配列の最後の列の文字列に１対１で対応しているので、ほぼ同程度の圧縮率の達成が期待できる。ブロックソート圧縮法の符号化方法によって、検索向きの圧縮法を提供することができる。

【００１８】

【発明の実施の形態】以下、本発明に係る各実施形態を、図１ないし図１１を用いて説明する。

【ブロックソート圧縮法】先ず、本発明のブロックソート圧縮データの検索法を説明する前に、図２および図３を用いて前提となるブロックソート圧縮法について説明

する。図２は、ブロックソート圧縮法の圧縮符号化過程を説明するための概略図である。図３は、ブロックソート圧縮法を具体的なデータにより説明するための図である。

【００１９】以下、本発明の実施形態では、一貫して、元のテキストが'ｃａｂｃｃａｂｃｃｃａｂｂｃａｂｃｃａｂｃａｃｃａｂｂｃａａａｂ'の３２個の文字からなるテキスト２００を圧縮した場合を例に採り説明していくことにする。

【００２０】ブロックソート圧縮法により圧縮して、符号化するときのアルゴリズムの概略は、以下のようになる。

【００２１】（圧縮－ステップ１）先ず、元のテキスト２００の巡回シフトをおこなって、全ての巡回シフト列２１０を求める。巡回シフトとは、元の列を一文字単位で右または左に、回転させてシフトさせることであり、図２の例では、テキスト２００を左側にシフトさせて、先頭からはみ出た文字'ｃ'が、最後尾にきている。

【００２２】この元のテキスト２００の例では、３２個の文字からなっているため、３２個の巡回シフト列２１０ができることになる。

【００２３】（圧縮－ステップ２）（圧縮－ステップ１）で生成された巡回シフト列を辞書式順序で整列させて、配列２２０を作る。

【００２４】（圧縮－ステップ３）配列２２０の最後の列１３０を取り出して、これを圧縮符号化する。このように元のテキスト２００からこのような手順を経て、最後の列１３０の列に変換することを、発案者の名前をとり、Ｂｕｒｒｏｗｓ－Ｗｈｅｅｌｅｒ変換、略してＢＷ変換と言っている。実際には、取ってくるのは配列２２０のどの列でも良いが、前記論文には、最後の列をとっている。

【００２５】また、この配列２２０の中で、元のテキストの位置番号２３０である'２５'についても圧縮しておく。

【００２６】元のテキスト２００とＢＷ変換した列は、長さは同じであるが同じ文字が続く傾向があることが知られているので、例えば、文字列の連続長を符号化することにより高い圧縮率が得られる。なお、ＢＷ変換した列の符号化の仕方は、外にもいろいろな方法が考えられ、必ずしも上記の方法にこだわる必要はない。

【００２７】このように、ブロックソート圧縮法は、辞書式順序で整列させた配列２２０に基づいて、符号化のためのデータを得るものなので、元のテキスト２００を圧縮するよりも効率の良い圧縮がおこなえるのでないかとして注目されているものである。

【００２８】次に、上記手順により、圧縮して符号化されたデータを復号して伸張する場合の手順を、図４および図５を用いて説明する。図４は、ブロックソート圧縮法の伸張復号化過程を説明するための概略図である。図５は、ブロックソート圧縮法を伸張復号化過程を具体的なデータにより説明するための図である。

【００２９】先ず、具体的な手順を説明する前に、図３に示されている整列位置番号と整列前位置番号について説明する。この整列位置番号と整列前位置番号は、ブロックソート圧縮法のアルゴリズムを理解する上において、非常に重要な概念である。

【００３０】整列位置番号は、巡回シフト列を辞書式順序に整列させた配列２２０の位置そのものである。

【００３１】整列前位置番号とは、ＢＷ変換した最後の列１３０を配列の最初の列に一致させるように、整列させたときに、整列させた文字が、整列する前にはどの整列位置番号の位置にあったかを示す位置番号である。

【００３２】具体的には、ＢＷ変換した最後の列１３０は、'ｃａｃｃａｃｃｃｃｃａａｂｂａａａａｂｃｃｃｂｂｃｂａｃｂｂｂ'である。

【００３３】これを整列させるために、先ず'ａ'の文字が来る。これは、整列前には、２番目にあったものだから、整列前位置番号は、０２となる。次に来るのも、'ａ'の文字である。次の'ａ'の文字が見出されるのは、ＢＷ変換した最後の列１３０で５番目なので、整列前位置番号は、０５となる。

【００３４】同様にして、'ａ'の文字が並び、'ｂ'の文字が最初に来るときの整列前位置番号は、１３になり、'ｃ'の文字が最初に来るときの整列前位置番号は、０１になる。このようにして、整列した列１３０が得られる。この対応の原理は、図４によって示されている。

【００３５】ここで、記号の約束をする。整列位置番号１４０と整列前位置番号１５０の対を、（整列位置番号，整列前位置番号）と書くことにする。例えば、'ａ'が最初に来るところは、（０１，０２）、'ｂ'が最初に来るところは、（１１，１３）、'ｃ'が最初に来るところは、（２０，０１）である。

【００３６】次に、上記手順により圧縮符号化したときのデータを、伸張復号化するときのアルゴリズムの概略は、以下のようになる。

【００３７】（伸張－ステップ１）（圧縮－ステップ３）で符号化した元のテキスト位置２３０と、ＢＷ変換した最後の列１３０を伸張復号化する。これを仮に第１段階の伸張復号化ということにする。この第１段階の伸張復号化は、もちろん、（圧縮－ステップ３）で符号化したアルゴリズムに基づくものである。

【００３８】これより、元のテキスト位置２３０である'２５'と、ＢＷ変換した最後の列１３０、'ｃａｃｃａｃｃｃｃａａｂｂａａａａｂｃｃｃｂｂｃｂａｃｂｂｂ'が得られたものとする。

【００３９】（伸張－ステップ２）（伸張－ステップ１）で得られたＢＷ変換した最後の列１３０を、辞書式で整列させる。そのとき、上記手順で得られるような（整列位置番号，整列前位置番号）の対も記憶してお

く。

【００４０】この例では、図３に示されるように（０１，０２）、（０２，０５）、（０３，１１）、…、（３２，２９）という対を得ることができる。

【００４１】（伸張－ステップ３）元のテキスト位置２３０、ＢＷ変換した最後の列１３０、整列させた列１６０および（整列位置番号，整列前位置番号）を基にして元のテキスト２００を復号する。これが第２段階の伸張復号化である。

【００４２】この例で示すと、以下のようになる。

【００４３】先ず、元のテキスト位置２３０が'２５'であるから、整列させた列１６０（図３の最初の列）の２５番目を見て、最初の文字、'ｃ'が復号される。この'ｃ'は、整列前には、８番目にあったことが、（２５，０８）の対を見ることにより分る（図５①）。次に、整列させた列１６０の８番目は、'ａ'である。もともと、この配列は巡回シフト列なのだから、この'ａ'は、最初の文字'ｃ'の次に来るものである。したがって、２番目の文字'ａ'が復号される（図５②）。

【００４４】同様に、（０８，１８）の対を見て、整列させた列１６０の１８番目に、'ｂ'が来ているので、３番目の文字'ｂ'が復号される。

【００４５】このような元のテキスト位置２３０から、'２５'から（２５，０８）、（０８，１８）、（１８，３１）、（３１，２６）、…というように連鎖的にたどっていくと、順々に、'ｃａｂｃ…'というように元のテキスト２００が復号化されて得られることになる。

【００４６】このようにブロックソート圧縮法は、巡回シフト列の性質を巧妙に利用して圧縮伸張をおこなうものである。

【００４７】〔ブロックソート圧縮データの検索方法の基本原理〕次に、図６を用いてブロックソート圧縮法により圧縮されたデータ（以下単に、「ブロックソート圧縮データ」という）に対して、特定のパータンを検索するための基本原理について説明する。図６は、本発明に係るブロックソート圧縮データの検索方法の検索過程を説明するための図である。

【００４８】本実施形態では、検索パターン１２０として、'ｃａｂｂｃａ'を取り上げる。この検索パターンは、元のテキスト２００には、２箇所に見出される。

【００４９】ここで、記号として検索パータンのｉ番目の文字を、Ｐ［ｉ］で表すことにする。この例では、図６にも示されているようにＰ［１］＝'ｃ'、Ｐ［２］＝'ａ'などである。

【００５０】アルゴリズムとしては、先ず、検索パータン１２０の最初の文字Ｐ［１］が、整列した列１６０（最初の列）のどこに見い出されかをサーチする。この整列した列１６０は、辞書式に整列しているため、その文字が最初に出現した場所から連続して出てくる個数だけ見れば良く、サーチは極めて容易である。元のテキスト２００から直接サーチする場合には、先頭の文字から始めて、順番に照合しなければならず、サーチが元のテキスト２００の長さ分だけおこなわなければならない。したがって、それと比較すると、最初から整列した列からサーチするためこの検索は極めて効率的であるということができる。

【００５１】図６の表では、Ｐ［１］＝'ｃ'の下に、２０～３２までの数字が並んでいるが、これが、整列した列１６０の整列位置番号である。実際に図３では、２０～３２に'ｃ'が来ていることが確認できる。

【００５２】次に、Ｐ［２］＝'ａ'を検索する。

【００５３】これは、各々Ｐ［１］で見つかった整列位置番号から整列前位置番号の対を求め、それにより検索する。すなわち、整列位置番号'２０'から整列前位置番号'０１'を求め、ブロックソート圧縮法の復号の原理により、'ａ'が復元されて、２文字目も一致することが分る。このように２文字目が'ａ'のパターンを調べると、図６の表の２列目から分るように、（整列位置番号，整列前位置番号）の組は、（２０，０１）、（２１，０３）、（２２，０４）、（２３，０６）、（２４，０７）、（２５，０８）、（２６，０９）、（２７，１０）となる。次の整列位置番号'２８'では、整列前位置番号'１８'であり、復号される文字は、'ｃ'となるため検索パターンと一致しないことが分る。そして、これ以降では、検索パターンと一致するパターンは原理上見出されない。というのも、配列２２０は、もともと辞書式に整列しているからである。

【００５４】同様に、次のＰ［３］＝'ｂ'を検索する。Ｐ［２］の整列位置番号から見出される候補は、（０３，１１）、（０４，１２）、（０６，１６）、（０７，１７）、（０８，１８）、（０９，１９）である。

【００５５】このようにして、Ｐ［１］～Ｐ［６］まで、一致するのは、図６に示されるように（２１，０３）、（０３，１１）、（１１，１３）、（１３，２０）、（２０，０１）の行６１０と（２２，０４）、（０４，１２）、（１２，１４）、（１４，２４）、（２４，０７）の行６２０の二箇所で一致することが分る。すなわち、この場所で検索する６個の文字が見つかったことを意味する。

【００５６】この検索法によれば、ブロックソート圧縮法の整列した列１６０と（整列位置番号，整列前位置番号）を基にして、先頭位置Ｐ［１］から順番に調べて行けば良く、しかも、各々の探索枝に対して、必ず検索パターンの長さだけ探せば良いので、元のテキスト列２００をサーチする場合に比べて極めて能率的な検索をおこなうことができる。

【００５７】〔該当箇所の前後の表示〕ところで、元のテキスト列２００から検索パターン１２０を検索するときに、その該当する箇所の前後の文字列を表示したいことが実用上よくある。

【００５８】この場合においても、本発明のブロックソート圧縮データの検索方法によれば、検索したときと、同様の手順により、該当箇所の前後の文字列を復号して表示することができる。

【００５９】例えば、上記の例で、図６の行６１０の箇所の前の文字列を表示したいとする。この場合には、整列前位置番号'２１'のときの整列位置番号を求めれば、Ｐ［１］の前の文字を求めることができる。すなわち、（整列位置番号，整列前位置番号）＝（ｘ，２１）にあたるｘを求めれば良い。ｘは２８となり、整列した列１６０の２８番目の文字が'ｃ'であることにより、求める文字は、'ｃ'であることが分る。その前の文字も同様にして、（ｘ，２８）から、ｘは１０となり、求める文字は、'ａ'であることが分る。

【００６０】逆に、図６の行６１０の箇所の後の文字列を表示したい場合には、一番最後の対（２０，０１）に注目して、整列位置番号'０１'のときの整列前位置番号を求めれば良い。すなわち、（整列位置番号，整列前位置番号）＝（０１，ｙ）にあたるｙを求める。ｙは０２であり、整列した列１６０の２番目の文字が'ａ'であることから、このＰ［６］のすぐ後ろの文字は、'ａ'であることが分る。同様に、（０２，ｙ）のｙを求めると、ｙは０５となり同様に次の文字も'ａ'であることができる。

【００６１】このように元のテキスト２００で検索パターン１２０がある所の前後の文字列は（整列位置番号，整列前位置番号）の連鎖をたどっていくことにより、自然と復号でき、これをＣＲＴやプリンタなどの出力装置に表示させることができる。

【００６２】〔あいまい検索への応用〕次に、図７を用いて本発明に係るブロックソート圧縮データの検索方法が、あいまい検索へも応用できることを説明する。図７は、本発明に係るブロックソート圧縮データの検索方法であいまい検索をおこなったときの検索過程を説明するための図である。

【００６３】テキスト検索において、いわゆる「あいまい検索」をおこないたいことが良くある。あいまい検索とは、例えば、単語の一部のみを指定し、その他の部分はなにが来ても良いとして一致するパターンを検索することである。例えば、'＊'の文字がドントケアの文字（ワイルドカードとも言う）を表すとして、検索パターンに'ａ＊ｂａｃ'が指定されたときには、全ての文字とマッチングするものと約束する。

【００６４】この例で、例えば、あいまい検索として、検索パターンとして'ｃａ＊＊ａｃ'が指定されたものとする。すなわち、Ｐ［３］＝Ｐ［４］＝'＊'で他の部分は、上の例と同様である。

【００６５】この場合には、先ず、既に述べた来た本発明の検索方法により、Ｐ［１］Ｐ［２］＝'ｃａ'の部分で一致する箇所を検索する。一致する部分は、（整列位置番号，整列前位置番号）の表現で表すと、図７に示される通り、（２０，０１）、（２１，０３）、（２２，０４）、（２３，０６）、（２４，０７）、（２５，０８）、（２６，０９）、（２７，１０）の８個である。

【００６６】次のＰ［３］Ｐ［４］＝'＊＊'の部分は、全ての文字とマッチングするので、そのまま、（整列位置番号，整列前位置番号）の連鎖をたどることになる。この過程では、候補は減らずに推移する。そして、この候補の中から後半のパターンＰ［５］Ｐ［６］＝'ｃａ'にマッチングするもののみを追跡し、最終的な絞り込みをおこなう。すなわち、次のＰ［５］のところでマッチングするのは、図のように５箇所であり、Ｐ［６］のところでさらに絞り込まれ、このあいまい検索の解としては図７に示されるように４箇所の場所が求まることになる。

【００６７】〔ブロックソート圧縮データの検索方法の効率化－その一〕上で、本発明に係るブロックソート圧縮データの検索方法の原理について述べたが、ここでは、図８を用いてさらに本発明の検索方法を効率的におこなうための工夫について説明する。図８は、元のテキスト列の出てくる個数に着目して復号と検索をおこなうための過程を説明するための図である。

【００６８】上記ブロックソート圧縮データの検索方法の原理では、ブロックソート圧縮法における第１段階の伸張復号化をおこなって、復号化されたＢＷ変換された列１３０（最後の列）に基づいて検索をおこなうものとして説明した。

【００６９】次に説明する本発明の検索方法は、必ずしも完全にＢＷ変換された列１３０を復号しなくても、検索をおこなえるものである。したがって、一層効率的な検索がおこなえることが期待される。

【００７０】この検索をおこなえるための条件は、元のテキスト２００の文字の出現個数がわかるように符号化されていることである。この例では、'ａ'が１０個、'ｂ'が９個、'ｃ'が１３個である。これから説明する検索方法の効率化のポイントは、文字の出現個数が分っているときには、ＢＷ変換した列１３０の先頭から順に処理していく毎に、整列位置番号と整列前位置番号の対を求めることができるため、先頭から順に復号して検索パターンとマッチング処理をおこなうことができる点にある。

【００７１】具体的に手順を追っていくと以下の通りである。先ず、ＢＷ変換された列１３０の最初の文字は、'ｃ'である。これは、'ｃ'の１番目であり、しかも、予め文字の出現個数が分っているので、'ｃ'の整列位置番号は、１０＋９＋１となる。すなわち、（整列位置番号，整列前位置番号）＝（２０，０１）が求まる。

【００７２】図８では、文字毎に整列前位置番号を並べたものであり、'ａ'の順番１のセルが整列位置番号が１、'ｂ'の順番１のセルが整列位置番号が１１、'ｃ'の

順番１のセルが整列位置番号が２０に該当することを示している。

【００７３】そして、次の文字'ａ'は、'ａ'の１番目であり、整列位置番号は、１である。すなわち、（整列位置番号，整列前位置番号）＝（０１，０２）である。同様に、３番目の文字は、'ｃ'については、整列位置番号は、１０＋９＋２＝２２であり、（整列位置番号，整列前位置番号）＝（２２，０３）である。これは、'ｃ'の順番２のセルの整列前位置番号が３であることに対応する。このように図８は、'ａ'、'ｂ'、'ｃ'の各文字が出てくるたびに、整列前位置番号を対応する行のセルに入れていけば、自動的に整列位置番号を求めることができることを示している。

【００７４】さて、検索パターン１２０は、'ｃａｂｂｃａ'であった。

【００７５】ここで、文字'ｂ'の最初のものが出てくるところまで、整列操作がおこなわれたとする。図８からすぐ見て取れるように、文字'ｂ'は、最初から１３番目に出てきて、すなわち、整列前位置番号は、１３であり、整列位置番号は、１０＋１＝１１である。

【００７６】ここまで、（整列位置番号，整列前位置番号）の対で、検索パターン１２０とマッチグするものは、（２１，０３）（０３，１１）（１１，１３）と、（２２，０４）（０４，１２）のシーケンスで、'ｃａｂｂ'と'ｃａｂ'までが照合できる。

【００７７】このようにして、ＢＷ変換した列１３０の２４番目の文字'ｂ'まで、整列操作がおこなわれた段階では、（２１，０３）（０３，１１）（１１，１３）（１３，２０）（２０，０１）と（２２，０４）（０４，１２）（１２，１４）（１４，２４）（２４，０７）で、検索パターン１２０の'ｃａｂｂｃａ'が照合できる。

【００７８】そして、これ以降では、もはや検索パターン１２０'ｃａｂｂｃａ'は、出現しないことが分る。これは、（整列位置番号，整列前位置番号）＝（１６，ｘｘ）にあたるｘｘには、文字'ｂ'にあたる整列前位置番号１１～１９が来ることとはない。これは、２４番目まで調べているので、他の場所に既に整列前位置番号１１～１９が使われていることが判明しているからである。したがって、これ以降には文字'ｈ'が来ることはなく、これ以降の照合操作はおこなう必要のないことがわかる。

【００７９】通常のテキストとの照合処理により、検索処理をおこなう場合には、最後の文字まで、照合しないと検索パターンを検出できないのと比較して、本発明のブロックソート圧縮データの検索方法の有利な点である。ただし、最悪の場合には最後まで整列操作をおこなわなければならない場合も生じうる。

【００８０】【ブロックソート圧縮データの検索方法の効率化－その二】次に、図１を用いて本発明の検索方法を効率的におこなうための他の工夫について説明する。図１は、本発明のブロックソート圧縮データの検索方法の概略手順を説明するための図である。

【００８１】これまで説明してきた本発明のブロックソート圧縮データ検索では、検索パターン１２０の先頭から照合処理をおこなってきた。しかしながら、検索パターン１２０の先頭文字Ｐ［１］のテキスト２００での出現個数が多いと、最初の照合処理が多くなり絞り込むまでの操作も多くなる。したがって、これを避けるためには、検索パターン１２０の文字列の中で、テキスト２００の中で出現個数の少ない文字を選び出し、その位置から照合処理を開始して、絞り込んだ後で、開始位置の前の文字を逆に遡って照合していくと効率的になる。

【００８２】また、検索パターン１２０の複数箇所の出現位置を同時に検出するよりも、先ず１箇所目を見つけた方が、２箇所目以降の絞り込みがはやくなる。

【００８３】検索パターン１２０'ｃａｂｂｃａ'の例では、'ａ'、'ｂ'、'ｃ'の三種類の文字が出てくるわけであるが、元のテキスト２００の中で、文字'ｂ'が９個で一番少ないので検索パターン１２０の３番目の文字'ｂ'に着目して検索を始める。

【００８４】図１に示されるように、前向きの照合では、（１１，１３）（１３，２０）（２０，０１）と照合されていき、後向きの照合では、（２１，０３）（０３，１１）というシーケンスで照合されていくことになる。このように検索パターンの任意の文字を選び出して、照合処理をおこなっていけるというのも、整列位置番号と整列前位置番号によって、前向きでも後向きでもまったく対称的に、復号ができるというブロックソート圧縮法の特徴によるものである。

【００８５】【ブロックソート圧縮法の修正した圧縮符号化】これまで、ブロックソート圧縮法を前提とする検索方法を述べてきた。この検索方法では、ブロックソート圧縮法の第１段階の伸張復号化をおこない、しかる後に（整列位置番号，整列前位置番号）により、第２段階の伸張復号化をおこなって検索パーンとの照合をおこなうものであった。第１段階の符号化の例としては、例えば、文字列の連続する長さで符号化する方法をあげた。

【００８６】ここでは、第１段階の符号化の段階で、（整列位置番号，整列前位置番号）の対を直接符号化することにより、２段階の符号化、復号化を１段階で済ませて、さらに効率的なブロックソート圧縮データに対する検索をおこなうアイデアについて説明する。

【００８７】以下、図９および図１０を用いてこれまでの例により説明することにする。図９は、本発明のブロックソート圧縮法の修正した圧縮符号化を説明するための図である。図１０は、圧縮符号化したデータを模式的に示した図である。

【００８８】ブロックソート圧縮法は、（整列位置番号，整列前位置番号）の対を利用して、第２段階の伸張

復号化をおこなうものであった。したがって、（整列位置番号，整列前位置番号）を直接に符号化すれば、復号時のこの対応付けを省略できるというが基本的な発想である。

【００８９】図９は、'ａ'のテーブル４１０、'ｂ'のテーブル４２０、'ｃ'のテーブル４３０毎に整列位置番号３４０と整列前位置番号３５０を対応してあげたものである。

【００９０】整列位置番号３４０も整列前位置番号３５０も０から始めている。これは、できるだけ符号化時に記憶容量を減らそうという技術的な工夫である。

【００９１】また、ＢＷ変換した列１６０は、おなじ文字が続く傾向があるため、整列前位置番号が同一の連番が続くことが期待される。したがって、整列前位置番号３５０は、各テーブルの相対位置であらわせば、圧縮率が高くなることが予想される。したがって、整列前位置番号３５０は、テーブルインデックス４４０と一緒に、各テーブルの相対番号として表現することにする。

【００９２】'ａ'のテーブル４１０の最初のエントリでは、整列位置番号３４０が００であり、整列前位置番号３５０が０１であり、テーブルインデックス４４０が'ａ'になっている。これは、図３の（整列位置番号，整列前位置番号）としては、（０１，０２）に該当する。また、'ａ'のテーブル４１０の３番目のエントリは、整列位置番号３４０が０２であり、整列前位置番号３５０が００であり、テーブルインデックス４４０が'ｂ'になっている。図８で示したように、'ｂ'のテーブルの最初の位置は、１１番目を表しているので、図３の（整列位置番号，整列前位置番号）としては、（０３，１１）である。

【００９３】これを実際に符号化するときには、整列前位置番号３５０と整列位置番号３４０の差を求めて相対的に符号化することにする。そして、これをテーブルインデックスと共に復号可能なように符号化する。

【００９４】このようにして符号化されたデータを符号化の仕方が良くわかるような書き方をすると、図１０のようになる。これは、テーブルインデックスと相対位置を符号化し、さらに、連続する文字の現れる場合の表記を工夫したものである。この表記で ｉ＋ｊ と書いてあるのは、ｉが ｊ 個連続していることを示している。

【００９５】この図１０で、ａ（１，３）は、テーブルインデックスが'ａ'で、整列前位置番号３５０と整列位置番号３４０の差３６０が、１と３、次のｂ（－２＋，０＋４）は、テーブルインデックスが'ｂ'であり、差３６０が－２，－２と続き、０が４つ続くことを意味している。

【００９６】〔ブロックソート圧縮データの検索方法のアルゴリズム〕最後に、図１を用いてこれまで説明してきたことを基にしてブロックソート圧縮データの検索方法のアルゴリズムを整理して述べることにする。図１は、本発明のブロックソート圧縮データの検索方法の概略手順を説明するための図である。

【００９７】元のテキスト２００がブロックソート圧縮法によって圧縮符号化されたデータが記憶媒体に記憶されているとする。

【００９８】そして、検索をおこなう検索パターン１２０が指定されているとする。

【００９９】本発明の検索方法は、圧縮符号化データ１００を伸張復号化しながら、検索パターン１２０との照合処理を可能にするものであった。

【０１００】検索は、任意の文字からおこなえるのであるが、元のテキストの出現個数の一番少ない文字、この例では'ｂ'からおこなうのが効率的である。復号化するに際しては、（整列位置番号１４０，整列前位置番号１５０）の対を順次連鎖的にたどって、検索パターンとマッチングするテキストの部分列を絞り込みながら、テキストの前後の双方向にわたって復号していく。第１段階の符号化で、出現する文字の個数がわかるようにしておくか、（整列位置番号１４０，整列前位置番号１５０）の対自体を符号化するようにすれば、圧縮符号化データ１００の全てを元のテキスト２００として復号しなくても順次、検索がおこなえることはこれまで述べきた通りである。

【０１０１】検索が成功して、該当箇所が見つかったら、必要に応じてマッチング箇所の前部と後部がどのように並びになっているかをユーザに提示してやる。

【０１０２】

【発明の効果】本発明のブロックソート圧縮データの検索方法は、検索対象の文字列パターンがテキスト中の数箇所に出現している場合でも、すべてを対象に先頭から同時に照合することができる。しかも、検索パターンの長さ分の照合を終了したところで、すべての照合可能な位置が検出される。そのため、効率的な高速検索ができるデータの高能率圧縮法である。また、検索パターンの前後の文字列の復号が直接おこなえるので、画面に検索検出位置の前後の文字列を同時表示することができるので応用面でも便利である。また、ブロックソート圧縮法の整列位置番号と整列前位置番号の対を直接符号化しておけば、この検索方法に有効に利用できる。

【０１０３】このように本発明によれば、ブロックソート圧縮法で圧縮されたデータに対してブロックソート圧縮法の特質を活かすことにより、全ての符号化されたデータを復号しなくても、逐次必要なデータだけ復号して検索することによって高速検索を可能にするブロックソート圧縮データ検索法を提供することができる。

【０１０４】また、その検索に適したブロックソート圧縮法の符号化方法を提供することができる。

【図面の簡単な説明】

【図１】本発明のブロックソート圧縮データの検索方法の概略手順を説明するための図である。

【図２】ブロックソート圧縮法の圧縮符号化過程を説明するための概略図である。

【図３】ブロックソート圧縮法を具体的なデータにより説明するための図である。

【図４】ブロックソート圧縮法の伸張復号化過程を説明するための概略図である。

【図５】ブロックソート圧縮法を伸張復号化過程を具体的なデータにより説明するための図である。

【図６】本発明に係るブロックソート圧縮データの検索方法の検索過程を説明するための図である。

【図７】本発明に係るブロックソート圧縮データの検索方法であいまい検索をおこなったときの検索過程を説明するための図である。

【図８】元のテキスト列の出てくる個数に着目して復号と検索をおこなうための過程を説明するための図である。

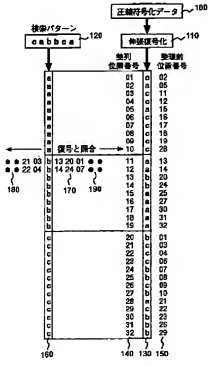
【図９】本発明のブロックソート圧縮法の修正した圧縮符号化を説明するための図である。

＊【図１０】圧縮符号化したデータを模式的に示した図である。

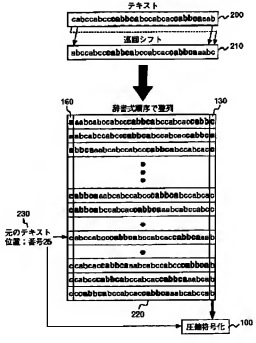
【符号の説明】

１００…圧縮符号化データ、１１０…第１段階の伸張復号化、１２０…検索パターン、１３０…ＢＷ変換された（配列の最後列の）列、１４０…整列位置番号、１５０…整列前位置番号、１６０…整列させた（配列の先頭列の）列、１７０…検索された整列番号、１８０…検索パターンの前部の復号化、１９０…検索パターンの後部の復号化、２００…元のテキスト、２１０…巡回シフト列、２２０…巡回シフト列の整列配列、２３０…元のテキストの位置番号、５００…第２段階の復号化、３４０…整列位置番号（００から始まる）、３５０…整列前位置番号（００から始まる）、３６０…整列前位置番号と整列位置番号の差、４１０…'ａ'テーブル、４２０…'ｂ'テーブル、４３０…'ｃ'テーブル、４４０…テーブルインデックス。

【図１】
Fig 1
図 1



【図２】
Fig 2
図 2



【図８】
Fig 8
図 8

順番	a	b	c
1	02	18	01
2	05	14	03
3	11	20	04
4	12	24	06
5	18	26	07
6	19	27	08
7	17	30	09
8	18	31	10
9	19	32	21
10	28		22
11			23
12			26
13			29

Fig 3

【図３】

図 3

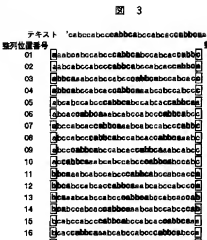


Fig 4

【図４】

図 4

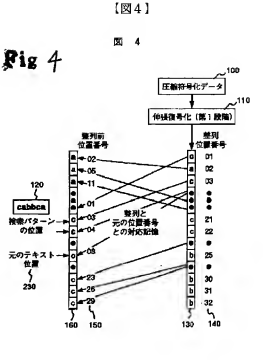


Fig 7

【図７】

図 7

P[1]	P[2]	P[3]	P[4]	P[5]	P[6]
c	a	*	*	a	a
20	01	02	05		
21	03	11	13	20	01
22	04	12	14	24	07
23	06	16	27		
24	07	17	30	32	09
25	08	18	31	26	00
26	09	19	32	29	
27	10	28	21		
28					
29					
30					
31					
32					

Fig 5

【図５】

図 5

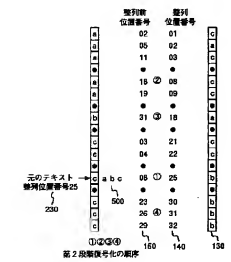


Fig 6

【図６】

図 6

P[1]	P[2]	P[3]	P[4]	P[5]	P[6]	
c	a	b	b	c	a	
20	01					
21	03	11	13	20	01	←610
22	04	12	14	24	07	←620
23	06	16				
24	07	17				
25	08	18				
26	09	19				
27	10					
28						
29						
30						
31						
32						

【図１０】

Fig 10

図 10

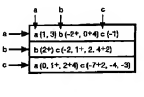
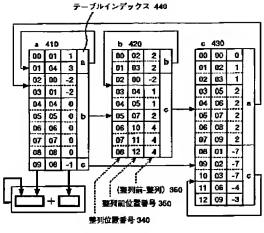


Fig 9

【図９】

図 9


テーブルインデックス 440
a 410
b 420
c 430
(整列前-整列) 350
整列前位置番号 360
整列位置番号 340